Agustín Caravá INFORME GITHUB – 6to 1ra

Nombre del archivo: newGameState_Caravá.py

En esta parte del código lo que se realiza es ingresarle al sistema el nombre del archivo de música que se va a utilizar para la música de fondo del juego.

```
#Instantiate mixer
mixer.init()

#Load audio file
mixer.music.load('marioSTAR.mp3')

print("music started playing....")

#Set preferred volume
mixer.music.set_volume(0.2)

#Play the music
mixer.music.play()
```

```
width, height = 400, 400

bg = 25, 25, 25

screen  = pygame.display.set_mode((height, width))
screen.fill(bg)

# Tamaño de nuestra matriz
nxC, nyC = 80, 80

# Estado de las celdas. Viva = 1 / Muerta = 0
gameState = np.zeros((nxC,  nyC))

#dimensiones de cada celda individual
dimCW = width / nxC
dimCH = height / nyC

xpos = 0
ypos = 0
bxpos = xpos+1
bypos = ypos

xvel = 0
yvel = 0
xtiempo = 0
ytiempo = 0
tiempo_global = 0

xpos_canon = xpos + 6
ypos_canon = ypos + 52

pauseExect = True
stay = True
```

En esta sección de código se ingresan al programa las variables nuevas y sus respectivos valores, como el tamaño de la matriz, y las variables de posición como xpos, ypos, bxpos, bypos.

En la línea 66 comienza el bucle de ejecución en donde estarán todos los comandos del juego.

Dentro del mismo :

```
for event in ev:
    # Detectamos si se presiona una tecla.
    if event.type == pygame.KEYDOWN:
        if event.key == pygame.K_LEFT:
            xvel = xvel - 1
        elif event.key == pygame.K_RIGHT:
            xvel = xvel + 1
        elif event.key == pygame.K_UP:
            yvel = yvel - 1
        elif event.key == pygame.K_DOWN:
            yvel = yvel + 1
        elif event.key == pygame.K_SPACE:
            gameState[xpos_canon,ypos_canon] = 2
            #Load audio
            mixer.music.load('oof_disparo.mp3')
            mixer.music.set_volume(0.2)
            #Play the music
            mixer.music.play()
        else:
            pauseExect = not pauseExect
    if event.type == pygame.QUIT:
        stay = False
        pygame.quit()
```

Nos encontramos con este bucle “for” que se encarga de detectar si apretamos alguna de las teclas configuradas.

En el caso de haber alguna tecla apretada se realizara la acción correspondiente, como mover la nave o disparar.

```
if (xpos != bxpos) or (ypos != bypos):

    #AVION
    #cuerpo
    gameState[bxpos+50,bypos+50] = 0
    gameState[bxpos+50,bypos+51] = 0
    gameState[bxpos+50,bypos+52] = 0
    gameState[bxpos+50,bypos+53] = 0
    gameState[bxpos+50,bypos+54] = 0
    gameState[bxpos+50,bypos+55] = 0
    gameState[bxpos+50,bypos+56] = 0
    gameState[bxpos+50,bypos+57] = 0
    gameState[bxpos+50,bypos+58] = 0
    gameState[bxpos+50,bypos+59] = 0
    gameState[bxpos+60,bypos+60] = 0

      #alas
    gameState[bxpos+51,bypos+53] = 0
    gameState[bxpos+49,bypos+53] = 0
    gameState[bxpos+52,bypos+54] = 0
    gameState[bxpos+48,bypos+54] = 0
    gameState[bxpos+53,bypos+55] = 0
    gameState[bxpos+47,bypos+55] = 0

    gameState[bxpos+51,bypos+59] = 0
    gameState[bxpos+49,bypos+59] = 0

    gameState[xpos+50,ypos+50] = 1
    gameState[xpos+50,ypos+51] = 1
    gameState[xpos+50,ypos+52] = 1
    gameState[xpos+50,ypos+53] = 1
    gameState[xpos+50,ypos+54] = 1
    gameState[xpos+50,ypos+55] = 1
    gameState[xpos+50,ypos+56] = 1
    gameState[xpos+50,ypos+57] = 1
    gameState[xpos+50,ypos+58] = 1
    gameState[xpos+50,ypos+59] = 1


      #alas
    gameState[xpos+51,ypos+53] = 1
    gameState[xpos+49,ypos+53] = 1
    gameState[xpos+52,ypos+54] = 1
    gameState[xpos+48,ypos+54] = 1
    gameState[xpos+53,ypos+55] = 1
```

Llegamos a una sección del código donde dibujamos dentro de la cuadrilla de juego la forma del avión deseado, mediante un método de punteo, llevando punto por punto cada uno con su ubicación mediante coordenas.

Tiene dos partes iguales el dibujo, la primera igualando cada posición de los puntos a 0, para borrar los puntos que están igualados a 1, a medida que se mueve la nave. Para evitar el arrastre.

```
for y in range(0, nxC):
    for x in range (0, nyC):

        #Movimiento de escombros
        if tiempo_global % 6 == 0:
            if (y in range(79))  and (gameState[x,y] == 3):
                gameState[x,y] = 0
                gameState[x,y+1] = 4
            if (y in range(79))  and (gameState[x,y] == 4):
                gameState[x,y] = 3



        #fisica del disparo
        if (y in range(79))  and (gameState[x,y+1] == 2):
            if gameState[x,y] == 3:
                gameState[x,y] = 0
                gameState[x,y-1] = 0
                gameState[x+1,y] = 0
                gameState[x-1,y] = 0
                gameState[x+1,y-1] = 0
                gameState[x-1,y-1] = 0
                hit_count += 1
            else:
                gameState[x,y] = 2

        if (y in range(79)) and (gameState[x,y] == 2) and (gameState[x,y+1] == 0
            gameState[x,y] = 0

        # Calculamos el polígono que forma la celda.
        poly = [((x)   * dimCW, y * dimCH),
                ((x+1) * dimCW, y * dimCH),
                ((x+1) * dimCW, (y+1) * dimCH),
                ((x)   * dimCW, (y+1) * dimCH)]

        # Si la celda está "muerta" pintamos un recuadro con borde gris
        if gameState[x, y] == 0:
            pygame.draw.polygon(screen, (40, 40, 40), poly, 1)
        # Si la celda está "viva" pintamos un recuadro relleno de color
        elif gameState[x, y] == 3:
            pygame.draw.polygon(screen, (200, 100, 0), poly, 0)
        elif gameState[x, y] == 2:
             pygame.draw.polygon(screen, (255, 255, 255), poly, 0)
        else:
            pygame.draw.polygon(screen, (200, 100, 100), poly, 0)
```

En la sección de código que se ve en la foto, se genera un bucle “for” para poner las condiciones de juego, como los disparos, con su física, la caída de los escombros y de más, como el color de cada uno de ellos, entre otros.